CLAIMS

We claim:

1. A method comprising:

receiving a list of candidate primitive actions comprising primitive actions related to a change in a file system state of a first file system, and primitive actions related to a change in a file system state of a second file system, both file systems constituting replicas of a shared file system; and

generating a schedule of non-conflicting primitive actions comprising one or more primitive actions from the list.

2. A method as recited in claim 1 wherein the generating a schedule comprises:

selecting a candidate primitive action from the list based on a value assessment of the candidate primitive actions.

3. A method as recited in claim 1 wherein the generating a schedule comprises:

selecting a candidate primitive action from the list based on a value assessment of the candidate primitive actions; and

executing the selected candidate primitive action starting from a checkpoint file system state; and

scheduling the selected candidate primitive action if the executing the selected candidate primitive is successful.

4. A method as recited in claim 1 further comprising:

receiving a log constraint representing a relationship between two of the primitive actions of the first file system, or between two of the primitive actions of the second file system.

5. A method as recited in claim 4 wherein the log constraint comprises a parcel relationship, a predecessor-successor relationship, or an alternatives relationship.

6. A method as recited in claim 1 further comprising:

receiving an object constraint representing a relationship between one of the primitive actions of the first file system and one of the primitive actions of the second file system.

7. A method as recited in claim 6 wherein the object constraint comprises a mutually-exclusive relationship or a best-order relationship.

8. A method as recited in claim 6 wherein the object constraint is based on application semantics related to the primitive actions of the shared file system.

9. A method as recited in claim 1 further comprising:

proposing the generated schedule to a user.

10. A method as recited in claim 1 further comprising:

committing the generated schedule on at least one of the first file system and the second file system.

11. A method as recited in claim 1 further comprising:

selecting one of the scheduled primitive actions to undo; and

undoing the selected scheduled primitive action.

12. A method as recited in claim 1 further comprising:

selecting one of the scheduled primitive actions to undo;

undoing the selected scheduled primitive action; and

undoing another of the primitive actions that depends on the selected scheduled primitive action to undo.

13. A method as recited in claim 1 wherein the receiving the list comprises:

decomposing a file system command at the first file system into the primitive actions related to a change in a file system state.

14. A method as recited in claim 1 wherein the generating comprises:

selecting a candidate primitive action from the list;

executing the selected candidate primitive action starting from a checkpoint file system state;

determining whether the executing the selected candidate primitive action was successful; and

if the executing the selected candidate primitive action was successful, adding the selected candidate primitive action to the schedule.

15. A method as recited in claim 14 wherein the generating further comprises:

if the executing the selected candidate primitive action was successful,

removing the selected candidate primitive action from the list,

removing from the list another candidate primitive action that may execute only before the selected candidate primitive action, and

removing from the list another candidate primitive action that conflicts with the selected candidate primitive action.

16. A method as recited in claim 14 wherein the generating further comprises:

if the executing the selected candidate primitive action was not successful,

removing the selected candidate action from the list, and

removing from the list all actions that are in a parcel with the selected candidate action.

17. A method as recited in claim 14 wherein the generating further comprises:

if the executing the selected candidate primitive action was not successful,

rolling back side effects that resulted from executing the selected candidate action, and

rolling back side effects that resulted from previously executing other actions that are in a parcel with the selected candidate action.

18. A method comprising:

receiving a first file system command;

receiving a second file system command;

decomposing the first file system command into one or more corresponding first primitive actions;

decomposing the second file system command into one or more corresponding second primitive actions;

receiving an object constraint indicating a relationship between a selected one of the first primitive actions and a selected one of the second primitive actions; and

if the object constraint indicates mutual exclusion, scheduling either the selected one of the first primitive actions or the selected one of the second primitive actions in a schedule of non-conflicting primitive actions based on the object constraint

otherwise, scheduling both the selected one of the first primitive actions and the selected one of the second primitive actions.

19. A method as recited in claim 18 further comprising:

executing the selected one of the first primitive actions and the selected one of the second primitive actions on a file system with the object

constraint to determine whether the selected first primitive action and the selected second primitive action conflict.

20. A method as recited in claim 18 further comprising:

logging the first primitive actions in a first action log;

logging the second primitive actions in a second action log;

combining the first action log and the second action log into a reconciliation log;

selecting a primitive action from the reconciliation log; and

executing the selected primitive action on a file system.

21. A method as recited in claim 20 wherein the selecting a primitive action comprises:

selecting a primitive action from the reconciliation log that has a higher value that the other primitive actions in the reconciliation log.

22. A method as recited in claim 18 wherein the relationship indicated by the object constraint comprises a mutually exclusive relationship.

23. A method as recited in claim 18 further comprising:

identifying a primitive action that conflicts with the scheduled primitive action among the first primitive actions and the second primitive actions; and

excluding the identified conflicting primitive action from the schedule of non-conflicting primitive actions.

24. A method as recited in claim 21 wherein the selecting a primitive action comprises:

attributing a higher value to a first primitive action than a second primitive action if scheduling the first primitive action would result in fewer conflicts with other primitive actions in the reconciliation log than would scheduling the second primitive action.

25. A processor-readable medium having processor executable instructions for performing a method comprising:

receiving a first file system command to change the state of a first file system;

generating a plurality of first primitive actions corresponding to the first file system command; and

receiving one or more log constraints representing a relationship between two of the plurality of first primitive actions; and

scheduling one or more of the first primitive actions in a non-conflicting schedule of primitive actions based on the one or more log constraints.

26. A processor-readable medium as recited in claim 25 wherein the method further comprises:

receiving a plurality of second primitive actions corresponding to a second file system command to change the state of a second file system;

selecting one of the first primitive actions and one of the second primitive actions;

receiving an object constraint representing a relationship between the selected first primitive action and the selected second primitive action; and

scheduling the selected first primitive action or the selected second primitive action based on the object constraint.

27. A processor-readable medium as recited in claim 25 wherein the one or more log constraints comprise user constraints.

28. A processor-readable medium as recited in claim 25 wherein the one or more log constraints comprise application constraints.

29. A processor-readable medium as recited in claim 25 wherein the method further comprises committing the schedule of non-conflicting actions on the first file system.

30. A processor-readable medium as recited in claim 26 wherein the object constraint depends upon application semantics.

31. A system comprising:

an input/output module receiving a file system command causing a tentative update to a file system state;

a reconcilable file system operable to receive the file system command, and generate a plurality of primitive actions representing the file system command and a log constraint representing a relationship between two primitive actions in the plurality of primitive actions; and

a log receiving the plurality of primitive actions and the log constraint.

32. A system as recited in claim 31 wherein the reconcilable file system comprises:

a decomposition module operable to decompose the file system command into the plurality of primitive actions; and

a recording module operable to receive the plurality of primitive actions and record the plurality of primitive actions in the log.

33. A system as recited in claim 32 wherein the decomposition module is further operable to generate the log constraint and communicate the log constraint to the recording module.

34. A system as recited in claim 32 wherein the log constraint is a user constraint.

35. A system as recited in claim 32 wherein the log constraint is an application constraint.

36. A system as recited in claim 32 wherein the log constraint is a parcel constraint indicating that the two primitive actions must be executed together.

37. A system as recited in claim 32 wherein the log constraint is a predecessor-successor constraint indicating that the two actions must be executed in a prescribed order.

38. A system as recited in claim 32 wherein the log constraint is an alternatives constraint indicating that only one of the two actions must be selected and executed.

39. A system as recited in claim 31 wherein the reconcilable file system is further operable to receive a schedule of non-conflicting primitive actions and commit the non-conflicting primitive actions to the file system.

40. A system as recited in claim 39 wherein the reconcilable file system is further operable to roll back changes that resulted in the tentative file system state in order to commit the schedule of non-conflicting primitive.